N93-11944

SYSTEM DIAGNOSTIC BUILDER

Joseph L. Nieten
GHG Corporation
1300 Hercules, Suite 111, Houston, TX 77058

Roger Burke
NASA,JSC
DK42, Houston, TX 77058

ABSTRACT

The System Diagnostic Builder (SDB) is an automated software verification and validation tool using state-of-the-art Artificial Intelligence (AI) technologies. Originally developed by GHG Corporation of Houston, Texas, the SDB is used extensively by project BURKE at NASA-JSC as one component of a software re-engineering toolkit. The SDB is applicable to any government or commercial organization which performs verification and validation tasks.

The SDB has an X-window interface, which allows the user to 'train' a set of rules for use in a rule-based evaluator. The interface has a window that allows the user to plot up to five data parameters (attributes) at a time. Using these plots and a mouse, the user can identify and classify a particular behavior of the subject software. Once the user has identified the general behavior pattern of the software, he can train a set of rules to represent his knowledge of that behavior.

The training process builds rules and fuzzy sets to use in the evaluator. These rules are built using a special implementation of the ID3 algorithm. The fuzzy sets classify those data points not clearly identified as a particular classification. Once an initial set of rules is trained, each additional data set given to the SDB will be used by a machine learning mechanism to refine the rules and fuzzy sets. This is a passive process and, therefore, it does not require any additional operator time.

The evaluation component of the SDB can be used to validate a single software system using some number of different data sets, such as a simulator. Moreover, it can be used to validate software systems which have been re-engineered from one language and design methodology to a totally new implementation.

Theory of Operation

The System Diagnostic Builder (SDB) uses an inductive machine learning technique to generate decision trees from data sets classified by a Subject Matter Expert (SME).

The primary objective of the SDB is to capture system knowledge from an SME. This process is known as knowledge acquisition. The knowledge must be represented in a manner to maximize usability. Representing the knowledge with a rule-base enables it to be used by a standard expert system shell or any other rule-based system. These rules could be used by a forward chaining system to detect faults and/or by a backward chaining system to identify the causes of these faults.

The SDB knowledge acquisition process is based on machine learning techniques combined with an X-window graphical interface. This interface allows an SME to train a rule-base from some number of data sets. The SME merely selects those parameters deemed to be pertinent to the identification of a particular system state. These parameters are then plotted on the screen, and with a click of the mouse button, the SME assigns some number of those data instances to a system state. This process is repeated until the entire data set is classified.

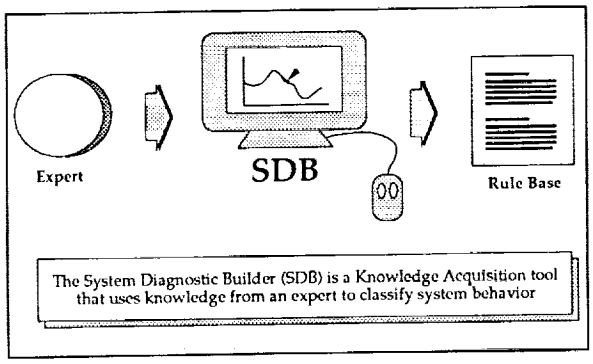

Figure 1

Those parameters used by the SME to classify system behavior must be available from some data stream within the system. However, the SME may require additional parameters during the classification process to insure accurate identification of the system's behaviors. These additional parameters will depend in some way on those basic parameters available from the system's data stream. The knowledge about these new parameters represents a whole new area within this knowledge acquisition scheme.

Additional knowledge about derived parameters is supplied to the SDB through the use of a standard ASCII text file which defines the relationships between actual system parameters and those parameters derived from the actual parameters. Future plans for the SDB include the incorporation of a prolog type dialog system to extract this Meta knowledge from the SME.

A temperature parameter is one example of a basic parameter used for training. Alone this parameter may not indicate very much, however, when used to calculate another key parameter, it becomes very useful. The knowledge about these calculations can either be generic to the type of science used by the system or explicitly added by the SME.

Once a data set is classified to the best of the SME's knowledge, it is submitted to an induction routine for generation of the rule-base. GHG has developed a special induction algorithm to generate the required rule-base. This new algorithm, called Turbo Induction, is a special implementation of the ID3 algorithm. In general, Turbo Induction enhances the discrete logic capabilities of ID3 with functions that map the continuous values of the system's parameters into discrete events. Specific capabilities of Turbo Induction are described later in this paper.

The SDB has an incremental learning capability. Multiple SME's can train the same data set, or a single SME can train multiple data sets without performing negative training. This is accomplished by managing the rule-bases from the X interface and by eliminating that SME knowledge which causes a conflict with existing knowledge. At this point, this process is done without consulting the SME. Future versions will provide a dialog capability to inform the SME of existing conflicts in the training domain for that system and request clarification from a 'super' SME.

Turbo Induction

Induction is a process where rules are automatically generated from a set of examples. ID3 is an induction algorithm developed by J.R. Quinlan. This algorithm analyzes data sets and recursively creates a decision tree. Each pass through the algorithm, a single attribute to be analyzed is chosen and a breakpoint value is calculated. This breakpoint is the largest inflection point. A node is added to the decision tree identifying the values on both sides of the breakpoint.

This approach depends heavily on the choice of attribute at each pass. A bad choice of attributes yields a set of rules that are unnecessarily complex. This approach also assumes that optimal decisions are made from existing data. There are no provisions for data provided by an SME during the operation.

While the ID3 algorithm works very well for discrete parameters and data, current implementations do not extend that success into the domain of continuous functions. Extremely large and noisy data sets produce a computational nightmare. For this reason, GHG Corporation has developed it's own implementation of the ID3 algorithm, with specific emphasis on training rules from very large and noisy data sets. This implementation is called Turbo Induction.

Several methods have been researched to eliminate the effects of noise on the induction process. One method took random samples from the data set and performed induction on only those data points. This method became less effective as the size of the data set grew. Analysis revealed that the density of those values which indicated particular paths through the decision tree became saturated, and actually diluted the other paths having less data points to support them. This is unacceptable when training a validation system.

Turbo Induction analyzes data sets using a pre-processor methodology. It performs localized induction and then maps those results into a final iteration that yields a set of discrete events. Thus, mapping the values from continuous functions into some number of discrete states (membership in a set). These sets are constructed and populated based on all knowledge about the system.

Turbo Induction has a procedure to add those parameters deemed important to the construction of a reliable rule-set. A superset of additional parameters must first be created. These parameters are analyzed during the pre-processor phase to determine their contribution to the final decision tree.

Applications

Currently, Project BURKE at NASA-JSC is using the SDB to Verify and Validate (V&V) the Shuttle Mission Simulator (SMS).

The SDB is used to train rule-bases for each single system within the SMS. Each of these systems has it's own data parameters to indicate it's behavior. An SME from each system trains a set of rules using data collected from the SMS. The SME also trains a set of rules using data collected from an actual flight or from another simulation which has already been certified (some kind of baseline).

Each of these rules is tested to determine it's accuracy level. In the event the generated rules are not accurate enough, the SME has the ability to refine the training and regenerate the rules. Once a valid set of rules exists for both the SMS and the baseline system, the sets of rules are combined to form a rule-base for the subject system.

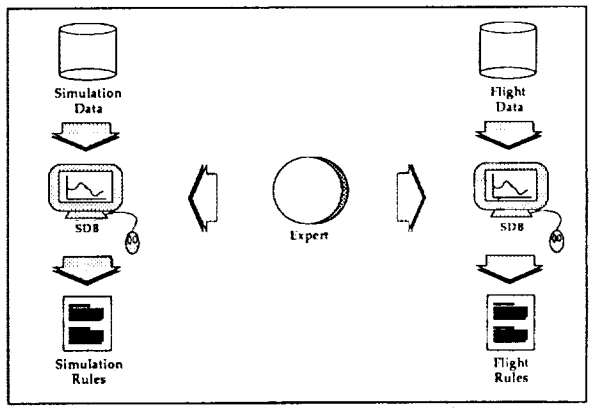

Figure 2

The verification and validation process is done passively. An operator can start a background process to observe the SMS data stream and classify the behavior of each data instance in real-time. This mode does not require an operator to monitor progress, only to respond to the warnings. The operator can also use data files to compare behaviors off-line. This approach can take more man power.

In short, an expert system shell is using the generated rule-base to identify the current state of the subject system. As each data instance is presented to the expert system, two rules should fire: One rule from the SMS rule set and one rule from the baseline rule set. The conclusion of both of these rules should agree. If they do not agree, then the expert system has identified an inconsistency.

Inconsistencies can be attributed to several reasons. The rule-sets may not agree due to an actual problem occurring in the baseline; which was then represented in the baseline rules set. In this case, the inconsistency is attributed to an anomaly that was not incorporated into the simulation. However, an inconsistency can also be an indication that the simulation has not modeled the real world properly. This situation is of particular interest for this application.

The SDB can be used in a number of Knowledge-Based applications. Each of these is discussed below.

As demonstrated in the SMS application, the SDB can be used to compare a software system's behavior to the behavior of some baseline system. The only requirement to perform this kind of analysis is data; data from both the software system to be analyzed and the baseline system.

Using the SDB to V&V re-engineered code is conceptually the same as the V&V process of a simulator. The only difference is the source of the baseline used for comparison. A subject for the baseline is already available: the original program source code. This makes the use of the SDB a perfect fit. The only difficulty may be in the visibility of variables within the original software.

Other applications can make use of the rule-bases generated by the SDB. The rule-base generated by the SDB represents actual 'behavioral' knowledge about the subject system. This knowledge is portable to any other rule-based system. In particular, these rules can be used in fault detection systems, fault isolation systems, and Intelligent Computer Aided Training (ICAT) systems.

One of the biggest draw backs for conventional expert systems is the time required to extract the knowledge from the expert. The SDB provides a vehicle for system knowledge to be captured one time and reused by any rule-based application.

Conclusion

The SDB provides a unique tool to perform knowledge acquisition for those systems with accessible data. The SDB also provides an excellent platform to perform verification and validation of conventional systems, using state-of-the-art technology.

REFERENCE

Holland, J.H., "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems", in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Quinlan, J.R., , "The Effect of Noise on Concept Learning," in *Machine Learning: An AI Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (EDs), Morgan Kaufman.

Quinlan, J.R., "Induction of decision trees", *Machine Learning 1*, 1.

Quinlan, J.R., "A case study of inductive knowledge acquisition", in *Applications of Expert Systems*, Quinlan J.R., Addison-Wesley.

Quinlan, J.R., "Generating production rules from decision trees", Proceedings 10th International Joint Conference Artificial Intelligence, Milan.

O'Keefe, R., "Simulation and Expert Systems - A Taxonomy and Some Examples", in *Simulation*, Society for Computer Simulation San Diego.